CS 162, Lecture 17: Friends and Operator Overload

9 May 2018
Complete Chess Demo
Operator Overload

• Able to overload the assignment operator, why not others?
• Overload: more than one definition for the same function name or operator, but with a different parameter listing
• Can overload most operators, cannot overload:
  • Scope resolution operator ::
  • .*
  • .
  • ?:
Example Operator Overload

class Circle {
    private:
        float radius;
        float center_x;
        float center_y;
    public:
        float get_radius();
        float get_center_x();
        float get_center_y();
        //insert mutators here
};

bool operator==(const Circle & c1, const Circle& c2) {
    if((c1.get_radius() == c2.get_radius()) && (c1.get_center_x() == c2.get_center_x()) && (c1.get_center_y() == c2.get_center_y()))
        return true;
    return false;
}
Friend

• Functions or classes declared with the friend keyword
• Non member function can access the protected and private members of the class if declared friend of that class
• Can have friend classes (not inheritance!) has access to the private and protected members of the friend
• Not a two way street: if declared friend in one class but not in the other the same access permissions are not provided
class Circle {
    private:
        float radius;
        float center_x;
        float center_y;

    public:
        float get_radius();
        float get_center_x();
        float get_center_y();
        friend bool operator == (const Circle &, const Circle &);
        //insert mutators here
};

bool operator==(const Circle & c1, const Circle& c2) {
    if((c1.radius == c2.radius) && (c1.center_x == c2.center_x) && (c1.center_y == c2.center_y))
        return true;
    return false;
}